Barcelona Supercomputing Center
Centro Nacional de Supercomputación

# SIMULATION-BASED PERFORMANCE ANALYSIS OF EC-EARTH 3.2.0 USING DIMEMAS

BSC-CES-2017-001
EC-Earth 3.2.0, Simulation, Computational Performance

X. Yepes-Arbós, M. C. Acosta, K. Serradell, A. Sanchez Lorente, F. J. Doblas-Reyes

Earth Sciences Department
*Barcelona Supercomputing Center - Centro Nacional de Supercomputación (BSC-CNS)*

12 January 2017

TECHNICAL REPORT

*Series: Earth Sciences (ES) Technical Report*

A full list of ES Publications can be found on our website under:

https://earth.bsc.es/wiki/doku.php?id=library:external:technical_memoranda

## Contents

# Index of figures

## Index of tables

## 1. Introduction

Earth System Models (ESMs) are complex applications executed in supercomputing facilities due to their high demand on computing resources. However, not all these models perform a good resource use and the energy efficiency can be well below a minimum acceptable.

One is EC-Earth [1], a global coupled climate model which integrates a number of component models to simulate the Earth system. The two main components are IFS 36r4 as atmospheric model and NEMO 3.6 as ocean model, both coupled using OASIS3-MCT. There are other small components like LIM3 as sea-ice model and runoff-mapper to distribute runoff from land to the ocean through rivers.

EC-Earth does not have a good computational performance [2]. For example, the scalability of this model using the T255L91 grid with 512 MPI processes for IFS and the ORCA1L75 grid with 128 MPI processes for NEMO achieves 40.3 of speedup, or 15.7 simulated years per day (SYPD). This means that the 81.2% of the resources are wasted. Moreover, it is not only to improve the computational efficiency of the current model, but also adapting it to support ultra-high resolution (UHR) experiments and to future exascale computers. Nevertheless, EC-Earth is not ready to scale at such levels of resolution.

For this reason it is necessary to apply different techniques to optimize correctly the model. But previously, a performance analysis is required to find the bottlenecks of the model. There are different approaches which have been done previously, from a very basic profiling analysis with gprof, to more in detail analyses such as tracing with Extrae and Paraver. These studies can be found [2][3].

Following this direction, in this study we will use a different approach based on simulation. We present the research done using the Dimemas simulator with EC-Earth. Using traces of the model collected with Extrae, Dimemas can simulate its message-passing behavior on a configurable parallel platform and extrapolate the impact of hardware changes in the performance of the application.

We will show the performance of EC-Earth in different scenarios: under the ideal machine, which has infinite bandwidth and no latency; the sensitivity of the application to parameters like infinite processor speed, network latency and network bandwidth; and detecting the slower model in the coupled system. The results obtained can help the model developers to anticipate the impact of a hardware change in their algorithm and help them to have a better comprehension of their codes.

The document is organized as follows. The experimental setup, including model description, environment and model configuration, is given in section 2. The tools and methodology applied are described in section 3 and **Error! Reference source not found.** respectively. Section 5 explains all the simulations done to analyze the performance of EC-Earth. Finally, section 6 contains the conclusions of the whole study.

# 2. Experimental setup

## 2.1. Model description

EC-Earth [1] is a global coupled climate model, which integrates a number of component models in order to simulate the earth system. It is developed by a consortium of European research institutions, which collaborate in the development of a new Earth System Model. The goal of EC-Earth is to build a fully coupled atmosphere-ocean-land-biosphere model usable for problems encompassing from seasonal-to-decadal climate prediction to climate change projections and paleoclimate simulations. It includes the following components:

- The **OASIS3-MCT coupler**: is a coupling library to be linked to the component models and whose main function is to interpolate and exchange the coupling fields between them to form a coupled system.

- The **Integrated Forecasting System (IFS)** as atmosphere model: is an operational global meteorological forecasting model developed and maintained by the European Centre of Medium-Range Weather Forecasts (ECMWF). The dynamical core of IFS is hydrostatic, two-time-level, semi-implicit, semi-Lagrangian and applies spectral transforms between grid-point space and spectral space. In the vertical the model is discretized using a finite-element scheme. A reduced Gaussian grid is used in the horizontal. The IFS cycle is 36r4 (as footnote).

- The **Nucleus for European Modelling of the Ocean (NEMO)** as ocean model: is a state-of-the-art modelling framework for oceanographic research, operational oceanography seasonal forecast and climate studies. It discretizes the 3D Navier-Stokes equations, being a finite difference, hydrostatic, primitive equation model, with a free sea surface and a non-linear equation of state in the Jackett. The ocean general circulation model (OGCM) is OPA (Océan Parallélisé). OPA is a primitive equation model which is numerically solved in a global ocean curvilinear grid known as ORCA. EC-Earth 3.2.0 uses NEMO's version 3.6 with **XML Input Output Server (XIOS)** version 1.0. XIOS is an asynchronous input/output server used to minimize previous I/O problems.

- The **Louvain-la-Neuve sea-Ice Model 2/3 (LIM2/3)**: is a thermodynamic-dynamic sea-ice model directly coupled with OPA.

- The **Hydrological extension of the Tiled ECMWF Surface Scheme for Exchange processes over Land (HTESSEL)** as land and vegetation module: is part of the atmosphere model. It solves the surface energy and water balance taking into account six different land tiles overlying a 4 layer soil scheme. The 6 tiles are: tall vegetation, low vegetation, interception reservoir, bare soil, snow on low vegetation, and snow under high vegetation.

- The **Tracer Model 5 (TM5)** as global chemistry transport model: describes the atmospheric chemistry and transport of reactive or inert tracers.

- The **runoff-mapper** component is used to distribute the runoff from land to the ocean through rivers. It runs using its own binary and coupled through OASIS3-MCT.

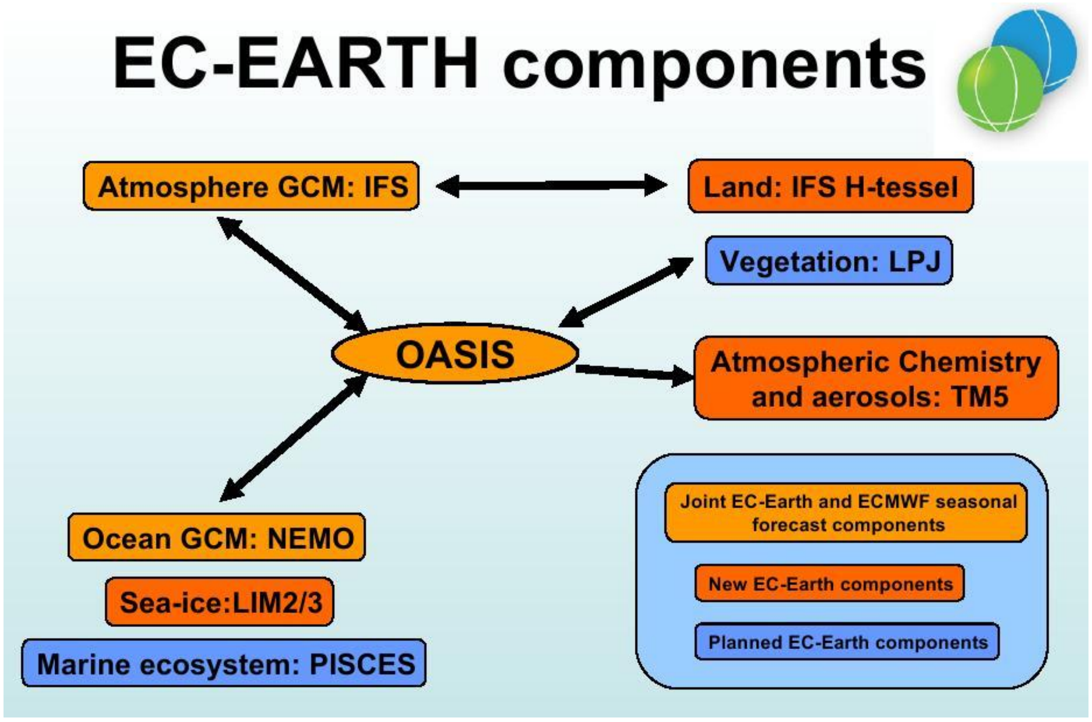


*Figure 1: Components used for the EC-Earth model*

The components are mainly written in Fortran, except XIOS which is written in C++. The model is parallelized using the message passing paradigm, particularly, the standard library MPI (Intel MPI v5) to be executed in distributed HPC clusters with several nodes.

For our simulation-based performance analysis, we will use IFS, NEMO, LIM3, XIOS and OASIS, being these components the essential part of EC-Earth and the modules mainly used for BSC scientists in the experiments.

## 2.2. Environment

All analysis were done on MareNostrum III, which is a supercomputer designed by IBM and it is hosted by the Barcelona Supercomputing Center (BSC). It has the next features:

- Peak performance of 1.1 PFLOPS
- 115.5 TB of main memory
- 2 PB of disk storage
- Main nodes:
  - 2x Intel SandyBridge-EP E5-2670/1600 20M 8-core at 2.6 Ghz
  - 128 nodes with 16x8 GB DDR3-1600 DIMMS (8 GB/core)
  - 128 nodes with 16x4 GB DDR3-1600 DIMMS (4 GB/core)

    - 2752 nodes with 8x4 GB DDR3-1600 DIMMS (2 GB/core)
    - 500 GB 7200 rpm SATA II local HDD
- Operating system: Linux - SuSe distribution
- LSF queue system
- Interconnection networks:
    - Infiniband FDR10
    - Gigabit Ethernet
    - 52 racks distributed in 120 m$^2$

## 2.3. Model configuration

We will run the 3.2.0 version of EC-Earth using the default configuration to compile and run the model on Marenostrum III, so we have the following conditions:

- EC-Earth 3.2.0. Revision 3051.
- MareNostrum III with 16 MPI processes per node. We use generic nodes with 32 GB and 16 cores, so we fill them up. The communications are done via Infiniband and the I/O is done via a Gigabit Ethernet network.
- Without check flags
- Default namelists values
- Mpirun order: IFS, NEMO, Runoff-mapper and XIOS.
- Use of an optimization to avoid mpi_allgather use a the northfold
- Fortran compilation flags: -O2 -fp-model precise -xHost -g -traceback -r8
- C compilation flags: -O2 -fp-model precise -xHost -g –traceback
- The software stack used to build and run the model includes:
- Compiler: Intel v13.0.1
- Intel MPI library v5.0.1.035
- MKL library v11.1.2
- NetCDF library v4.3.2
- HDF5 library v1.8.12-mpi
- GRIB library v.1.14.0

Furthermore, we have the following specific parameters:

| | |
|---|---|
| **Model** | Nemo-3.6 |
| **Configuration name** | ORCA1L75_LIM3 |
| **Resolution** | ORCA1L75 |
| **Modules** | OPA and LIM3 |
| **Time step** | 2700 seconds (32 time steps per day) |
| **Compilation keys** | key_trabbl key_vvl key_dynspg_ts key_ldfslp key_traldf_c2d key_traldf_eiv key_dynldf_c3d key_zdfddm key_zdftmx key_mpp_mpi key_zdftke key_lim3 key_iomput key_oasis3 key_oa3mct_v3 |

*Table 1: Configuration parameters for NEMO*

| | |
|---|---|
| **Model** | ifs-36r4 |
| **Resolution** | T255L91 |
| **Time step** | 2700 seconds (32 time steps per day) |

*Table 2: Configuration parameters for IFS*

| | |
|---|---|
| **Component model** | OASIS3-MCT |
| **Coupling frequency** | 2700 seconds (default value) |

*Table 3: Configuration parameters for OASIS3-MCT*

# 3. Tools

In previous studies [2][3] related to the performance analysis of EC-Earth, we studied its performance by using traces. Tracing is the process of recording event-based performance data along the execution of a program. Using a viewer we can see the behavior of the application in our machine, focusing on hardware counters, communication pattern or memory hierarchy. The tools used to trace the model were Extrae and Paraver, which are open-source and developed by the BSC Performance Tools group [4].

Furthermore, there is another BSC tool called Dimemas which is used in this report to perform a more advanced performance study of EC-Earth. The tools are briefly detailed in next subsections.

## 3.1. Extrae

Extrae is a package used to instrument the code automatically and/or manually through its API. It generates Paraver trace-files with hardware counters, MPI messages and other information for a post-mortem analysis. It can be downloaded and installed in any HPC facility.

## 3.2. Paraver

Paraver is a trace browser that can show a global visual qualitative perspective of the application behavior for later focus on the bottlenecks with a detailed quantitative analysis. The tool allows to create views with any parameter recorded and points to a region of a code, making process of modification easier.

## 3.3. Dimemas

Dimemas is a performance analysis tool for message-passing programs. It enables the user to develop and tune parallel applications on a workstation, while providing an accurate prediction of their performance on the parallel target machine. The tool is mainly command line but provides a GUI to create machine files in an easier way.

The simulator (Figure 2) takes as input a trace file of an execution and a target architecture file, and sequentially (the process can take long depending on the size of the input trace) produces another trace as output. Using Paraver, both traces can be opened to visually compare the changes produced by the simulation.

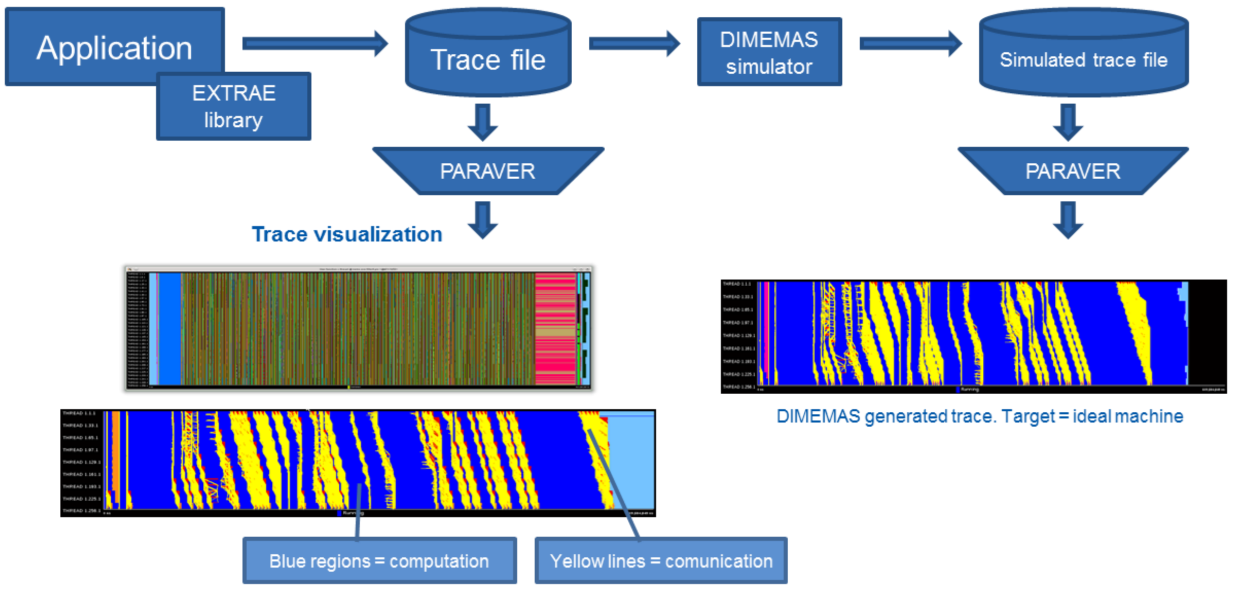


*Figure 2: Overview of a simulation-based performance analysis with Extrae, Paraver and Dimemas*

## 3.4. Collaboration

All the work presented has been done in collaboration with the Performance Tools team at the Barcelona Supercomputing Center. Indeed, for the developers of the tool, working with a huge and complex application helps them to debug and improve it. Particularly, before we started the simulation-based performance analysis with Dimemas, some workarounds had to be implemented in order to use the tools with EC-Earth:

- Version 3.4.1 of Extrae had a bug that made fail EC-Earth's tracing. It was detected and solved by BSC tools team.
- Applications can only be linked to C or Fortran library versions of Extrae at the same time, depending on the source code of the application to be traced. However, since EC-Earth is a hybrid application with both C and Fortran binaries, BSC tools team had to prepare an Extrae version for us to be able to trace both languages at the same time.
- Dimemas does not support intercommunicators yet, so we had to perform some small changes of the original trace file in order to simulate new traces.

These workarounds increased the technical level of the analysis and in consequence, delayed the report. In the other hands, to solve these issues Earth and Performance Tools got a very valuable experience, which have a great impact in future analysis and works.

# 4. Methodology

The experiment consists in tracing four time steps of EC-Earth using 512 MPI processes for IFS and 128 for NEMO. The decision of choosing this combination is explained in detail in section 5.1. As we previously commented in section 3, this is done by linking Extrae to the model without having to instrument the source code. Through an XML file we can set which performance metrics we want to generate. In this study we are mainly interested in MPI events, as it is the main goal of the performance study with Dimemas.

Once we have a trace, we can visualize it with Paraver to determine the behavior. After that we can simulate with Dimemas different target machine parameters to evaluate the new behavior of EC-Earth. This process is iteratively repeated as many times as different target machine parameters we want to simulate.

In particular, there are performed the following simulation tests:

- Parameter adjustment for MareNostrum III: a target architecture file is set to produce a simulated trace as close as possible to the behavior of the original trace. This trace is used to compare with all the other simulated traces.

- Ideal machine: the execution is simulated in a perfect node interconnection network with infinite bandwidth and no latency.

- Model sensitivity: several executions are simulated by changing processor speed, network bandwidth and network latency. These parameters are useful to study workload balancing, types of messages and communication patterns.

- Limiting model due to coupling: alternatively, one model is “disabled” (IFS or NEMO) by making it extremely faster in order to study the performance of the other model (NEMO or IFS). This is for studying the coupling impact over the two models and to know the slowest one.

They are explained and analyzed in section 5.3.

# 5. Simulation-based performance analysis

## 5.1. Selection of the appropriate case of simulation

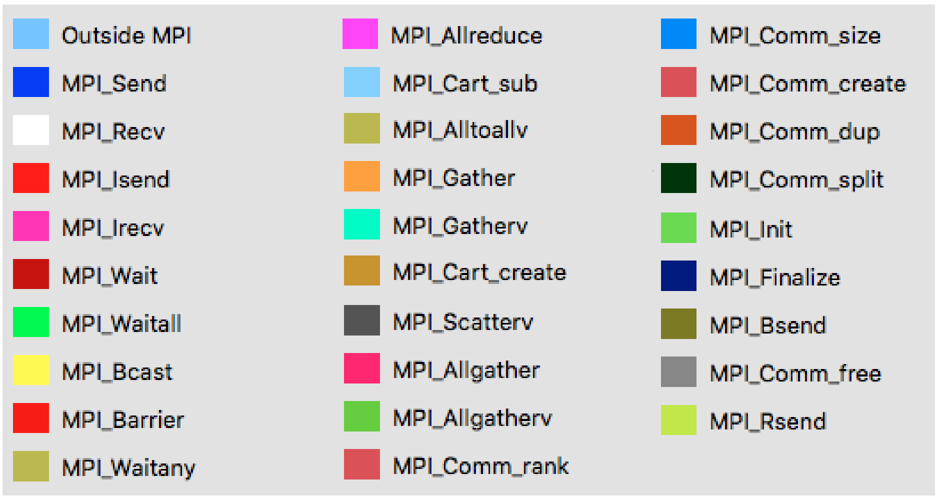
The first phase of the simulation-based performance analysis consists in selecting the appropriate case of study, i.e., to decide how many MPI processes will be used to simulate executions.

The ideal case would be to work with small executions, such as using 64 MPI processes for IFS and 16 for NEMO. This clearly enables to perform faster simulations, since traces' size is considerably low. Nevertheless, this scenario is not realistic at all, as we proved in a previous work [2] (section 4.3.1) that important inefficiencies only appear in executions with a high amount of MPI processes for IFS.

This happens because in the coupling process of EC-Earth via OASIS3-MCT, at the end of each time step is applied an expensive conservative method over a set of variables sent from IFS to NEMO. In this process, the IFS master process has to send a point-to-point message to the rest of IFS processes and then it receives a message from each one of them. Thus, this part of the code is completely serialized and increases with the number of processes, so the larger the number of IFS processes, the larger is the execution time of the conservative part.

Mainly for this reason and some other issues, it is important to use a realistic number of MPI processes for the simulations, although they slow down the simulation analysis.

Therefore, we will perform all the simulations using 512 MPI processes for IFS and 128 for NEMO. In Figure 4 there is the base trace that we will use. Each color represents an MPI function, which are detailed in Figure 3. In particular, the first color of the list (light blue) is not an MPI function, it simply means non-MPI code.



| | | | | | |
|---|---|---|---|---|---|
| 🟦 | Outside MPI | 🟪 | MPI_Allreduce | 🟦 | MPI_Comm_size |
| 🟦 | MPI_Send | 🟦 | MPI_Cart_sub | 🟥 | MPI_Comm_create |
| ⬜ | MPI_Recv | 🟨 | MPI_Alltoallv | 🟧 | MPI_Comm_dup |
| 🟥 | MPI_Isend | 🟧 | MPI_Gather | 🟩 | MPI_Comm_split |
| 🟪 | MPI_Irecv | 🟩 | MPI_Gatherv | 🟩 | MPI_Init |
| 🟥 | MPI_Wait | 🟧 | MPI_Cart_create | 🟦 | MPI_Finalize |
| 🟩 | MPI_Waitall | ⬛ | MPI_Scatterv | 🟫 | MPI_Bsend |
| 🟨 | MPI_Bcast | 🟥 | MPI_Allgather | ⬛ | MPI_Comm_free |
| 🟥 | MPI_Barrier | 🟩 | MPI_Allgatherv | 🟩 | MPI_Rsend |
| 🟨 | MPI_Waitany | 🟥 | MPI_Comm_rank | | |

*Figure 3: Legend with the colors representation of the MPI calls used by Paraver*

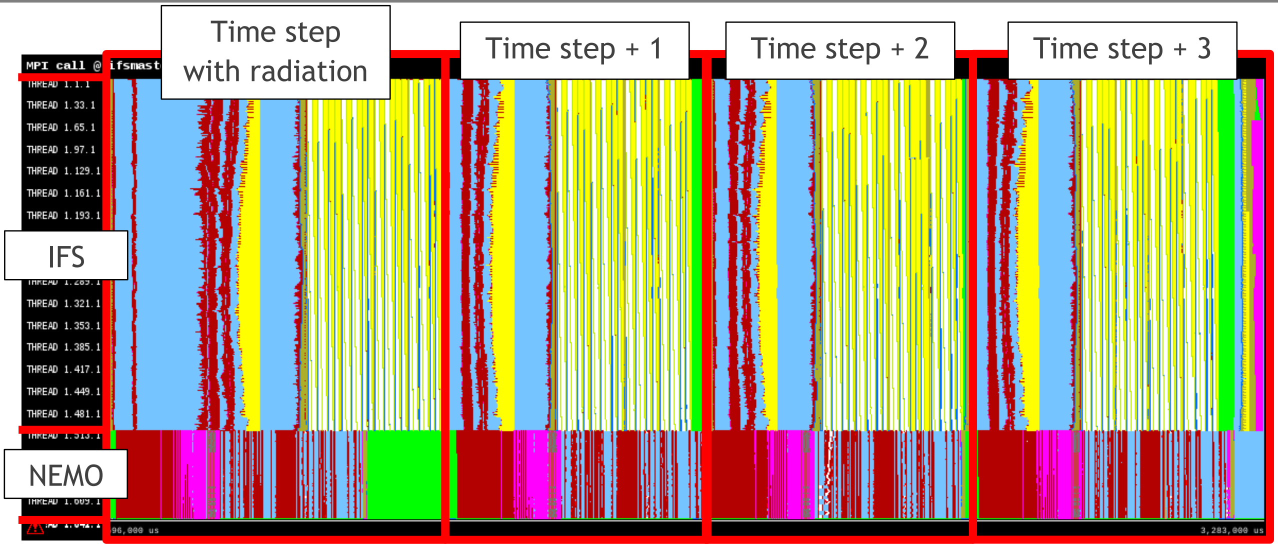


*Figure 4: View of the base trace that is used for all the simulations*

This trace consists of four EC-Earth time steps, which is the default repetitive pattern, as one out of four time steps in IFS executes radiation. In NEMO, all time steps are equal. The Figure 4 also describes the MPI processes that correspond to IFS and NEMO. In addition, there are two MPI processes at the end executing runoff-mapper and XIOS components. They are not highlighted in the trace because are not visible.

It is notable the huge amount of communications that are needed in both IFS and NEMO, but in many cases, they are not efficient. The simulation-based analyses will not only reveal performance communication details, but also computation details such as workload imbalances.

## 5.2. Simulation parameter adjustment for MareNostrum III

Once we have decided the number of MPI processes and before starting the Dimemas simulations, we have to adjust for MareNostrum III the simulation parameters. This means that we have to set the right Dimemas configuration to produce a simulated trace as close as possible to the original one.

After that, we will be able to make small changes to the Dimemas base configuration to analyze different test scenarios, such as the ideal machine, which consists in simulating a perfect interconnection network with infinite bandwidth and no latency.

In Figure 5 there is the original trace with an execution time of 2.687 seconds. Analogously, Figure 6 shows the same execution but generated with the Dimemas base configuration using the parameters that we estimated for MareNostrum III. This configuration is summarized in **Error! Reference source not found.**.

| Parameter | Value |
|---|---|
| Network bandwidth | 500 MB/s |
| Latency | 8 µs |
| Input / Output links per node | 6 / 6 |
| Processor speed ratio | 1.0 |
| Number of buses | 0 (no contention) |
| Process mapping | Fill nodes |
| Eager limit | 32K |

*Table 4: Dimemas base configuration*

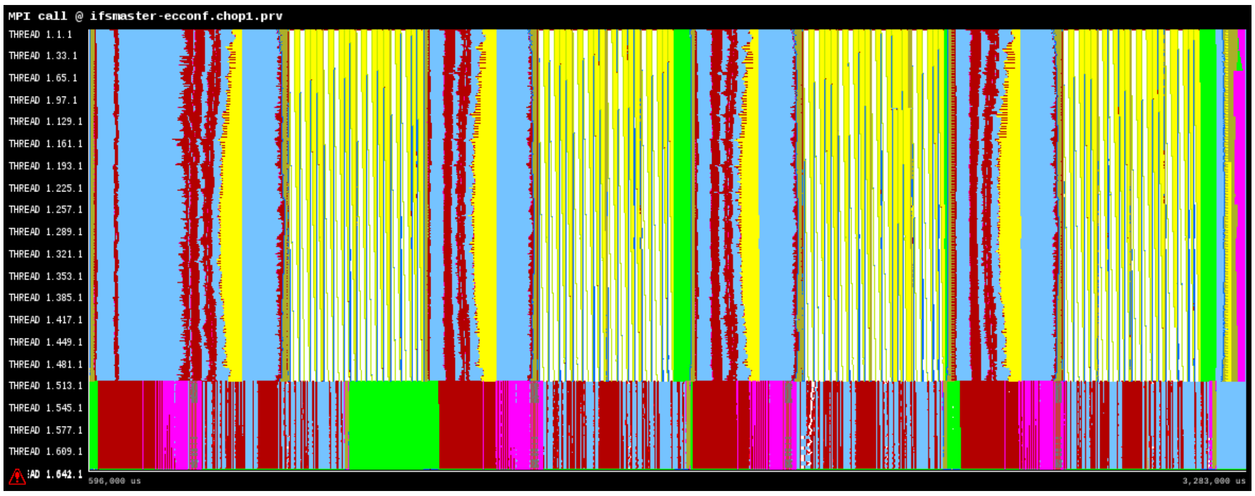


*Figure 5: Original trace*

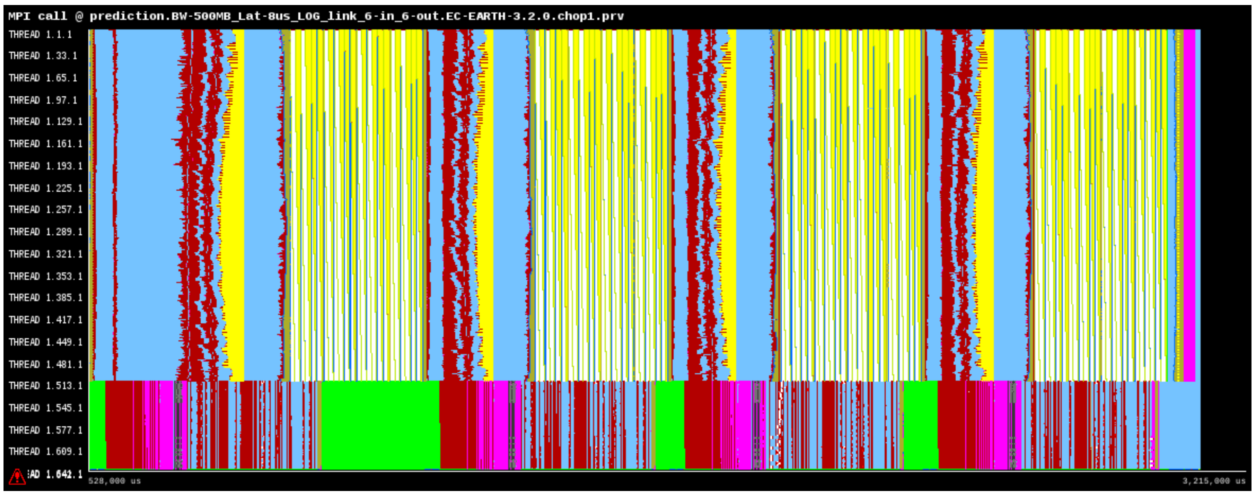


*Figure 6: Simulated trace with Dimemas base configuration*

As can be noticed in two previous figures, both traces are close, but not equal. Different issues make difficult to obtain exactly the same results since real environments have some problems such as shared network among third jobs, lost messages, etc., which cannot be considered in the simulated experiment. The simulated execution time is 2.574 seconds, slightly better than the original one.

We will stay on this configuration and takes as valid these results to proceed our analysis.

## 5.3. Simulation tests

Once we have a base configuration, we can proceed to simulate different test scenarios. Basically, there are performed three tests: ideal machine, model sensitivity and limiting model due to coupling. They are explained in the following subsections.

### 5.3.1. Ideal machine

The first test consists in simulating the ideal machine, which has a node interconnection network with infinite bandwidth and no latency. This configuration is interesting to analyze the new behavior when communications will not impact on performance. Thus, it is possible to see if applications have a good workload balance.

Figure 7 and Figure 8 show a comparison between the simulated base trace and the simulated ideal machine trace respectively. In both cases, the time window is the same to see the improvement of the ideal machine. Time is reduced from 2.574 to 2.044 seconds, suggesting that the 20.59% of execution time is communication, which is an overhead of parallel executions. It reduces the parallel efficiency of the model.

On the other hand, the simulated trace shows the inter-model workload imbalance of IFS and NEMO, but also individually. There are a lot of waiting regions, as a result of an inefficient work distribution. Basically, there are MPI_Wait (red) and MPI_Waitall (green). Moreover, there are collective calls, MPI_Bcast (yellow) and MPI_Allreduce (magenta), that are contributing to the inefficiency of the model.

Finally, there is another source of inefficiency, which are data dependencies. Not all the waiting calls are due to workload imbalances, but also to chains of data dependencies between MPI processes.

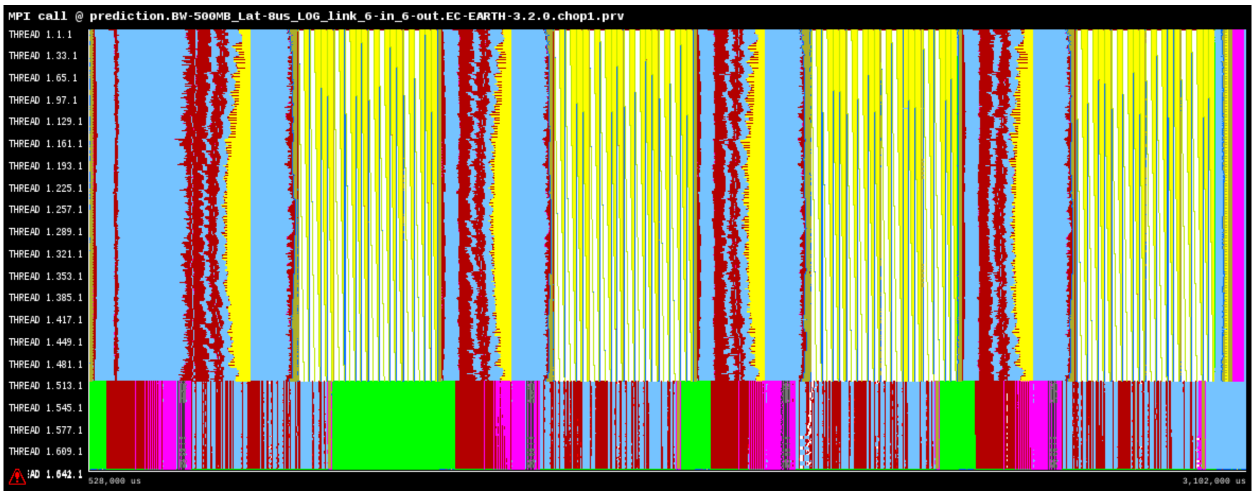


*Figure 7: Simulated trace with Dimemas base configuration*

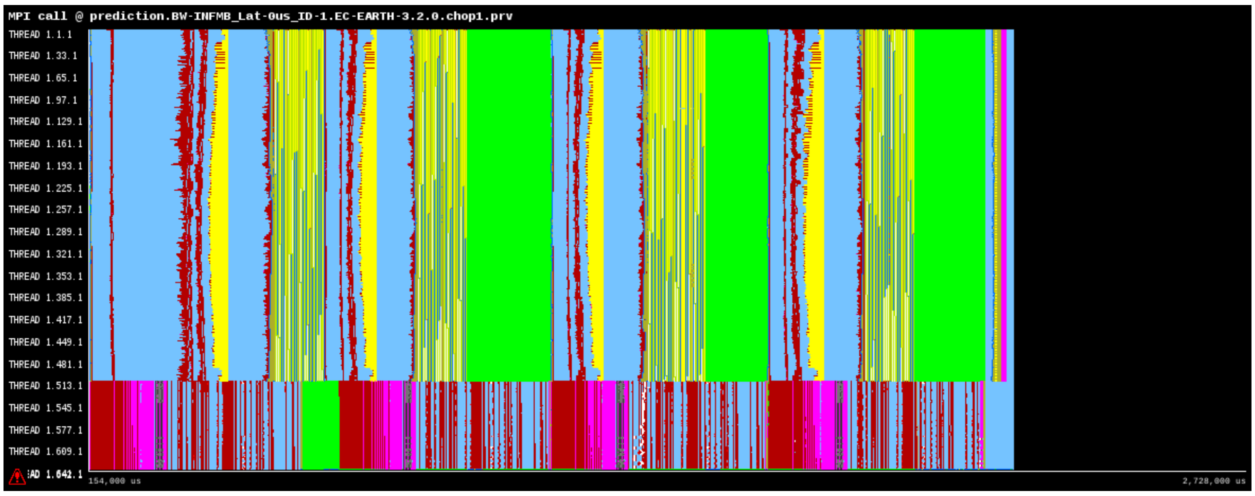


*Figure 8: Simulated trace with an ideal machine*

### 5.3.2. Model sensitivity

Another approach consists in testing the communication sensitivity. It is useful to know the impact of communication over the model. One way to proceed is to simulate infinite processor speed in order to determine the communication's overhead.

More precisely, as in the ideal machine case, in that scenario is possible to identify communication's overhead and data dependencies, because although computation chunks are removed, the time dedicated to transfer data is kept.

Thus, if there are regions of the trace that are reduced considerably is due to computation, whereas regions that remain almost equal is probably due to communications or chains of data dependencies.

In Figure 9 and Figure 10 there is a comparison between the simulated base trace and infinite processor speed respectively. It is visible that NEMO time steps are considerably reduced, but in IFS the reduction is not so evident, especially in the conservative part since the code is serialized (chain of data dependencies). This type of code does not benefit from faster processors.

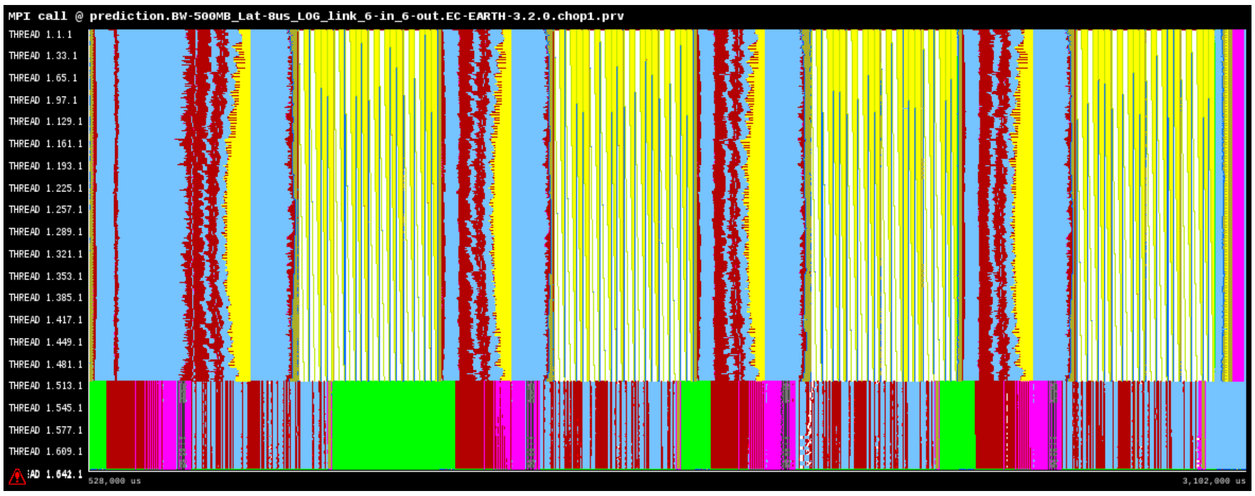


Figure 9: Simulated trace with Dimemas base configuration

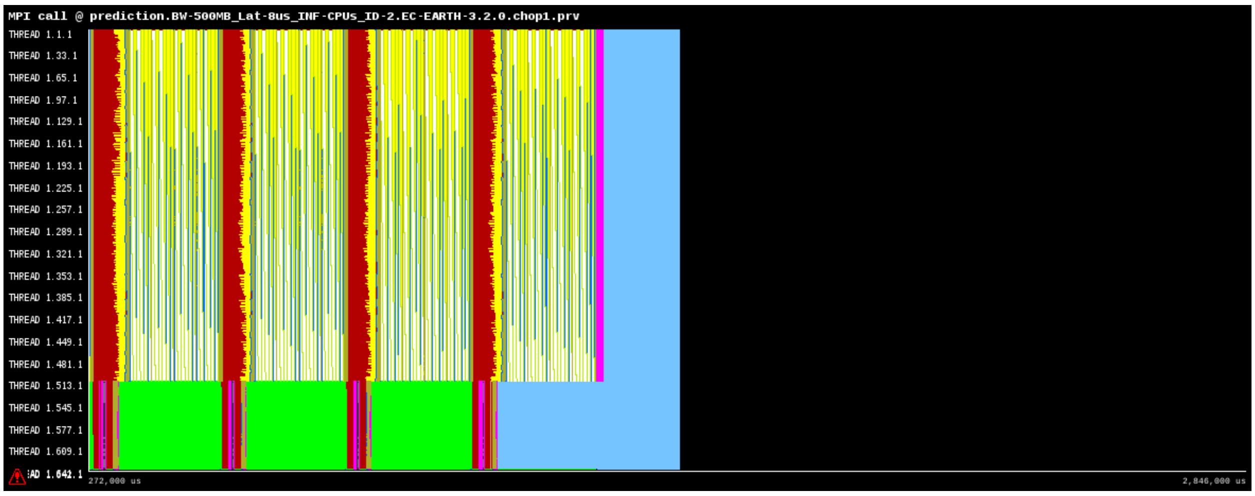


Figure 10: Simulated trace with infinite processor speed

The second way to test the communication sensitivity is changing the simulated values of latency or bandwidth. This is useful to determine the efficiency of communications. It is preferable to have few big messages rather than many small messages in order to exploit better the network bandwidth and reduce the latency of different messages.

For example, detecting small messages is possible by changing latency, whereas to detect big messages is possible by changing bandwidth. The values used in the sensitivity analysis of EC-Earth are the following:

- Latency: 1, 2, 4, 8, 16 µs
- Bandwidth: 100, 250, 500, 1000, 2000 MB/s

In both cases, values are changed in one parameter while fixing the other to the original value. This is, if we try different values for latency, then we fix bandwidth to the original adjusted value, 500 MB/s; if we try different values for bandwidth, then we fix latency to the original adjusted value, 8 us.

##### *5.3.2.1. Latency*

Figure 11, Figure 12, Figure 13, Figure 14 and Figure 15 show respectively simulations for 16, 8, 4, 2 and 1 µs of latency. The other parameters are equal for all the tests. Note that Figure 12 is the Dimemas base configuration trace. The execution times of the four time steps in each case are:

- 16 us: 3.407 seconds
- 8 us: 2.574 seconds
- 4 us: 2.278 seconds
- 2 us: 2.18 seconds
- 1 us: 2.145 seconds

Analyzing and comparing the traces and the execution times, it is deducible that the most benefiting part of the code is the conservative part in IFS. In fact, it is almost the only reduced part because it consists of many little and serialized messages that take advantage of better latencies. This reasoning corresponds to the one of the infinite speed processor simulation.

On the other hand, the execution time only improves considerably up to 8 µs of latency, because IFS regular time step (without radiation) is slower than NEMO time step. However, with 4 µs or less of latency, the overall improvement is lower, since the NEMO time step is slower than the IFS regular time step. Basically, the small improvement comes from the reduction in the IFS radiation time step (it is still slower than a NEMO time step).

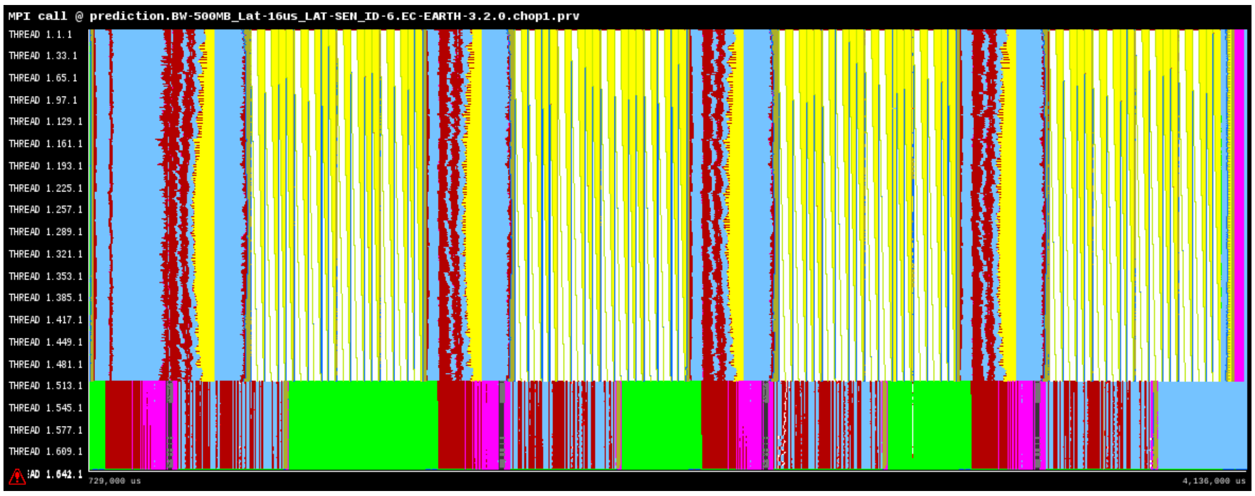


*Figure 11: Simulated trace with 16 μs of latency*

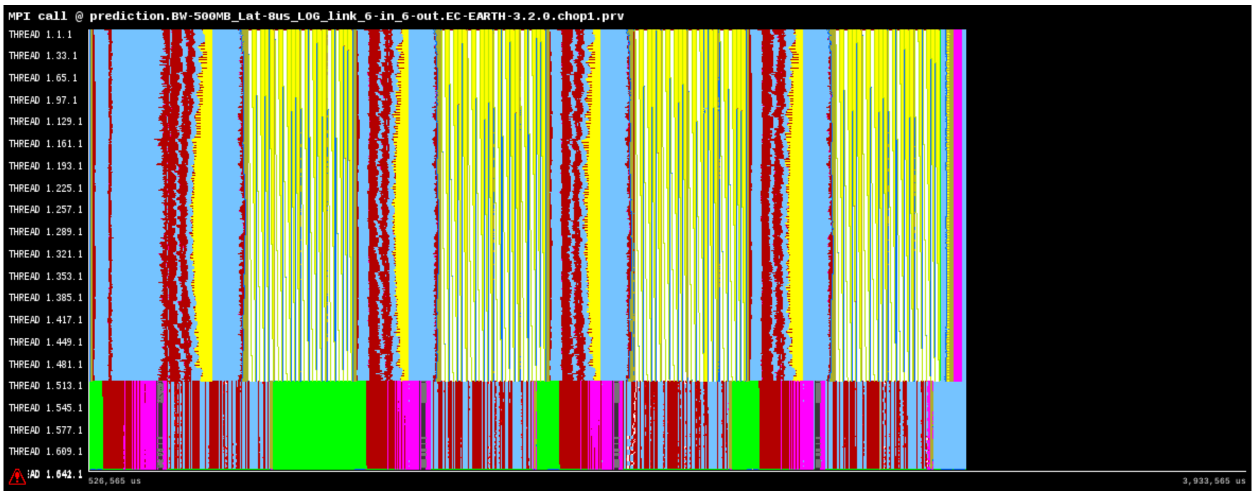


*Figure 12: Simulated trace with 8 μs of latency (Dimemas base configuration)*

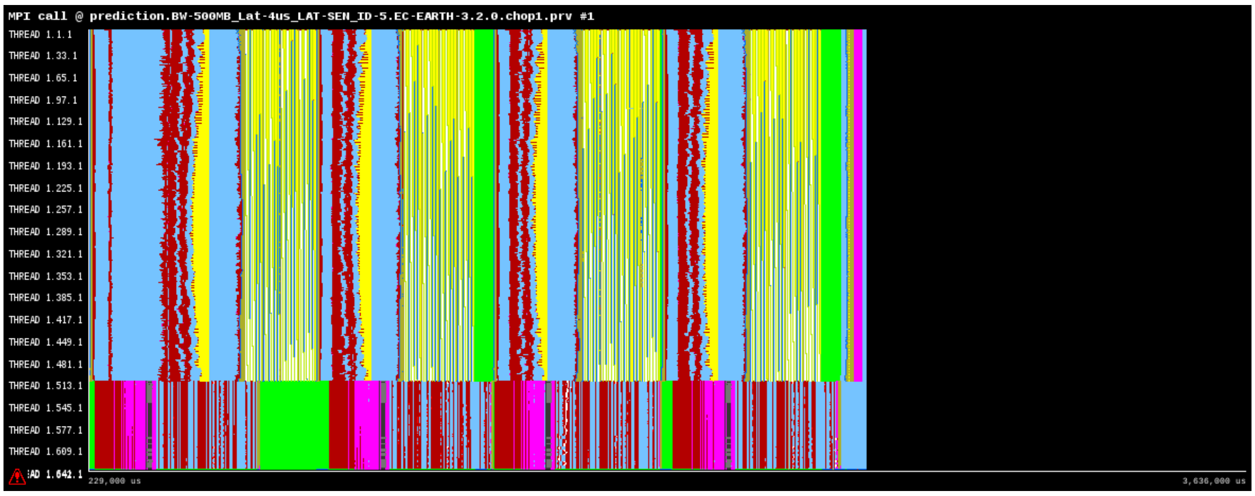


*Figure 13: Simulated trace with 4 μs of latency*

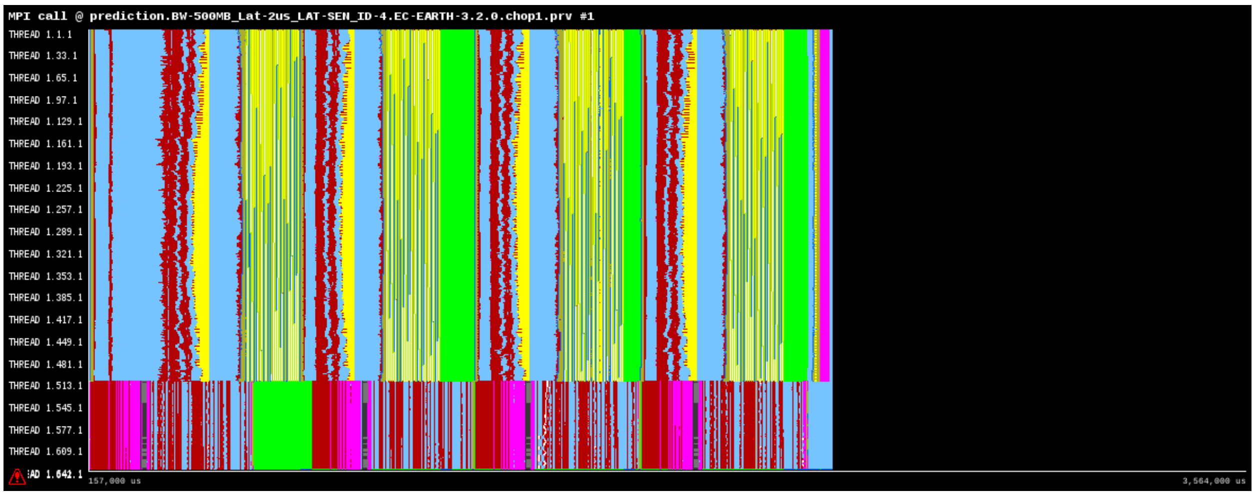


*Figure 14: Simulated trace with 2 μs of latency*

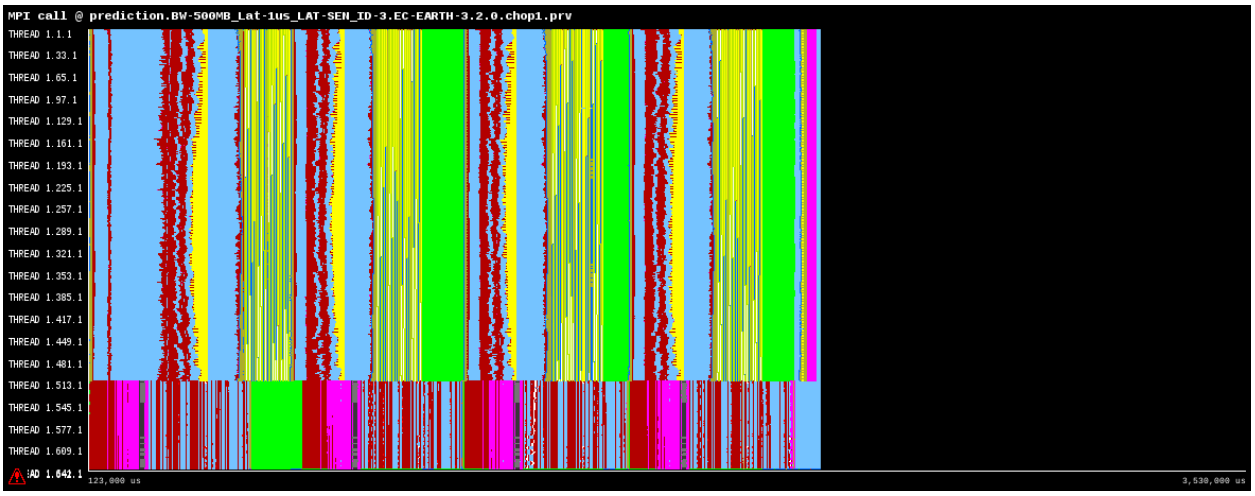


*Figure 15: Simulated trace with 1 μs of latency*

##### *5.3.2.2. Bandwidth*

Figure 16, Figure 17, Figure 18, Figure 19 and Figure 20 show respectively simulations for 100, 250, 500, 1000 and 2000 MB/s of bandwidth. The other parameters are equal for all the tests. Note that Figure 18 is the Dimemas base configuration trace. The execution times of the four time steps in each case are:

- 100 MB/s: 3.652 seconds
- 250 MB/s: 2.839 seconds
- 500 MB/s: 2.574 seconds
- 1000 MB/s: 2.465 seconds
- 2000 MB/s: 2.437 seconds

Analyzing and comparing the traces and the execution times, it is deducible that the model with more than 500 MB/s of bandwidth almost does not improve, since the node interconnection network has enough capacity and there is no contention.

However, with less than 500 MB/s of bandwidth, there are changes in the execution time. As can be seen in the traces, NEMO behavior almost does not change, but in IFS there is a clear region of MPI_Wait's (red) in each time step that requires an important bandwidth. Therefore, there are few big messages that maximize the use of the network.

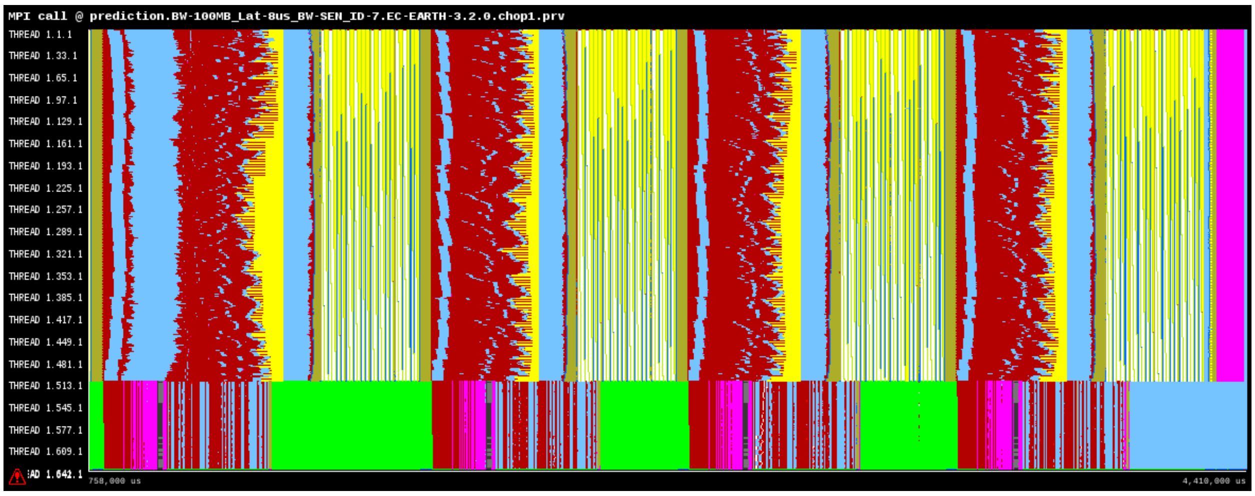


*Figure 16: Simulated trace with 100 MB/s of bandwidth*

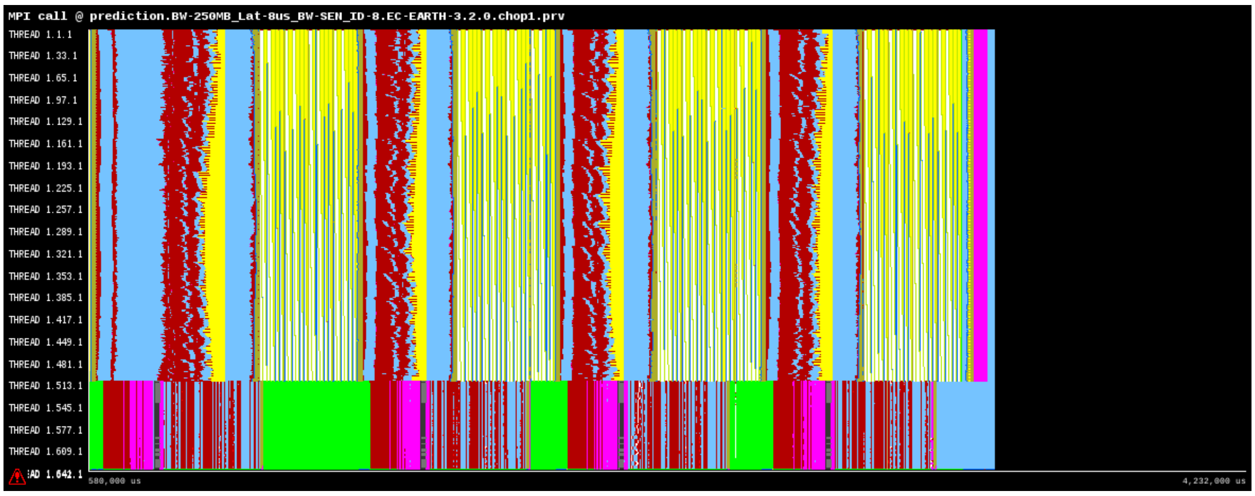


*Figure 17: Simulated trace with 250 MB/s of bandwidth*

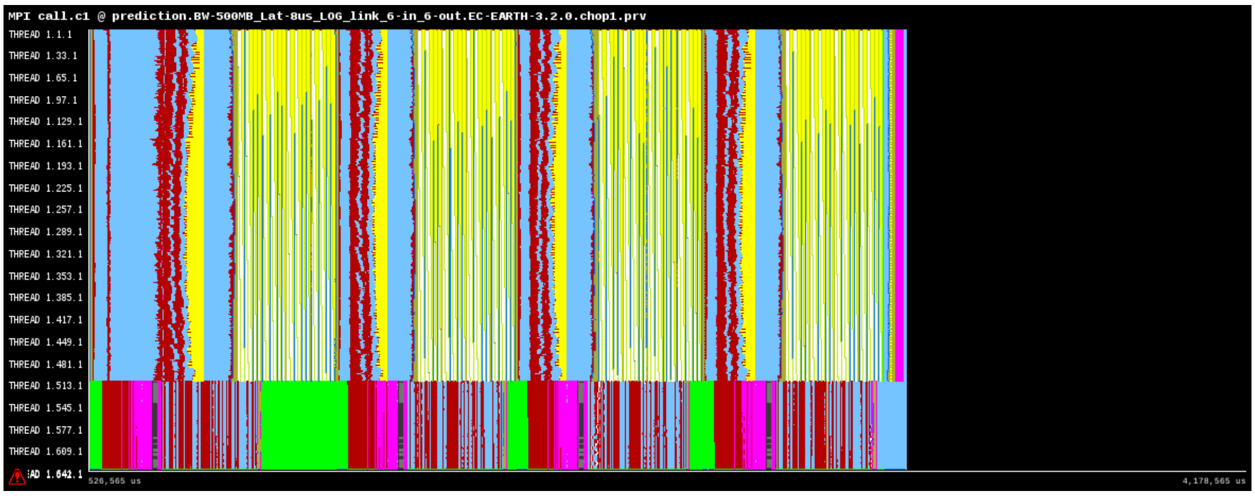


*Figure 18: Simulated trace with 500 MB/s of bandwidth (Dimemas base configuration)*

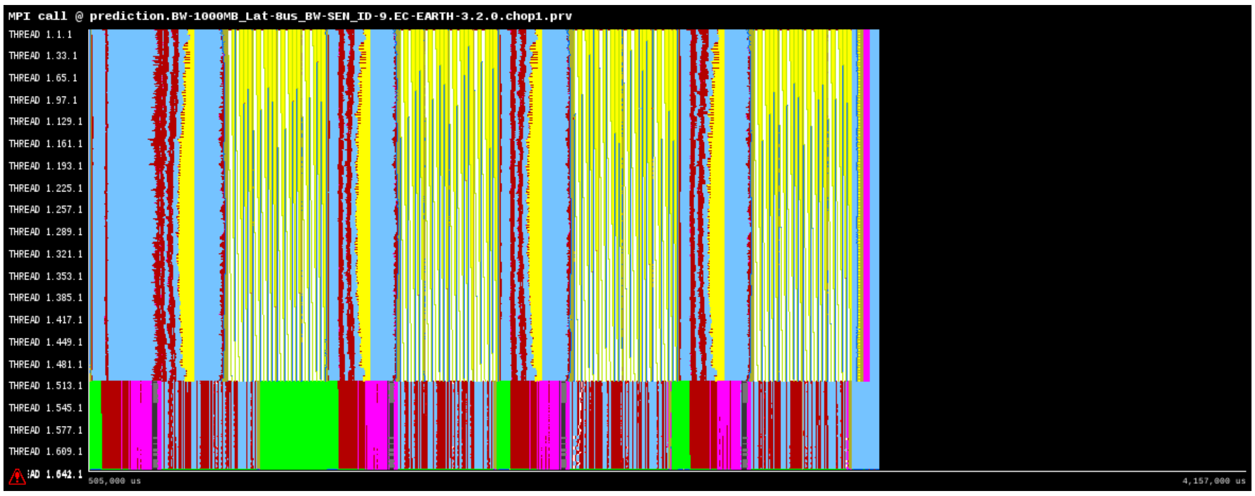


*Figure 19: Simulated trace with 1000 MB/s of bandwidth*

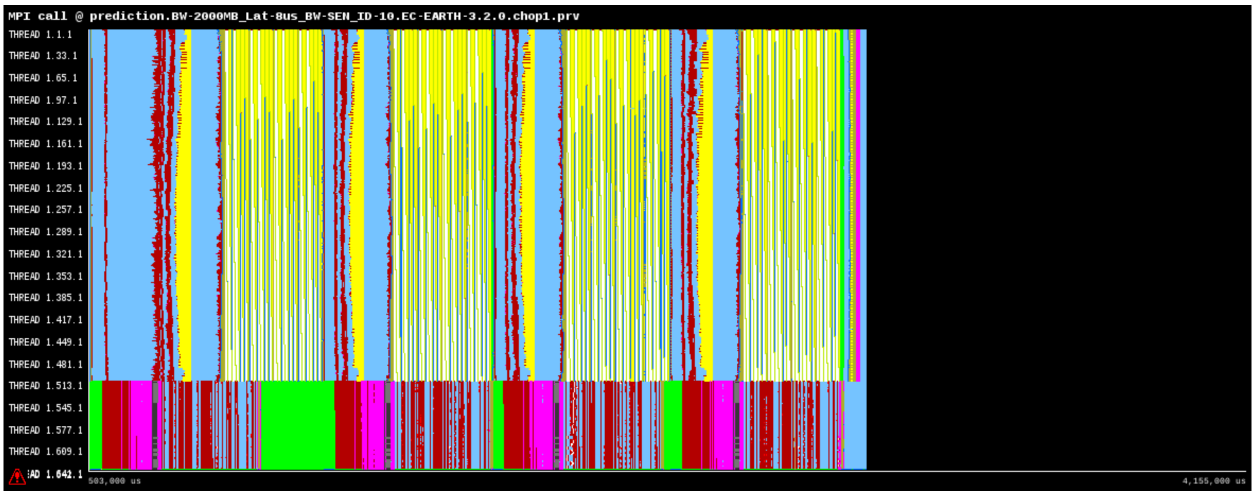


*Figure 20: Simulated trace with 2000 MB/s of bandwidth*

### 5.3.3. Limiting model due to coupling

The last simulation is devoted to see the impact of the coupling process between IFS and NEMO. The idea is to “disable” one of the models to see how coupling affects to the other one. By “disabling” is meant to make one of the models much faster than usual and not changing any parameter of the other one.

To make one model much faster can be achieved by simulating infinite processor speed and no latency for the nodes that execute the “disabled” model.

Figure 21, Figure 22 and Figure 23 show the simulated base trace, IFS “disabled” and NEMO “disabled” respectively. They use the same time window with regard to the first trace. The execution times are these:

- Simulation base trace: 2.574 seconds
- IFS “disabled”: 2.052 seconds
- NEMO “disabled”: 2.574 seconds

Analyzing and comparing the traces and the execution times, it is deducible that the slowest model for this combination of MPI processes (512 IFS, 128 NEMO) is IFS, since the traces of Figure 21 and Figure 23 have the same duration. In fact, NEMO could go faster as it is appreciable in Figure 22, but both models have to synchronize in each time step in order to exchange data through the coupler.

On the other hand, since to “disable” a model we are using infinite processor speed and no latency, but keeping the same bandwidth (parameter at the cluster level), the residual execution time that remains in each time step of the “disabled” model is due to transferred data. This means that it is only accounted time devoted to transfer data through the network. In NEMO, there is enough bandwidth with 500 MB/s to transfer all the messages (Figure 23), but in IFS there is not enough bandwidth, so for this reason the residual execution time in each time step (Figure 22) is larger than in NEMO. This is coherent with what was explained previously.

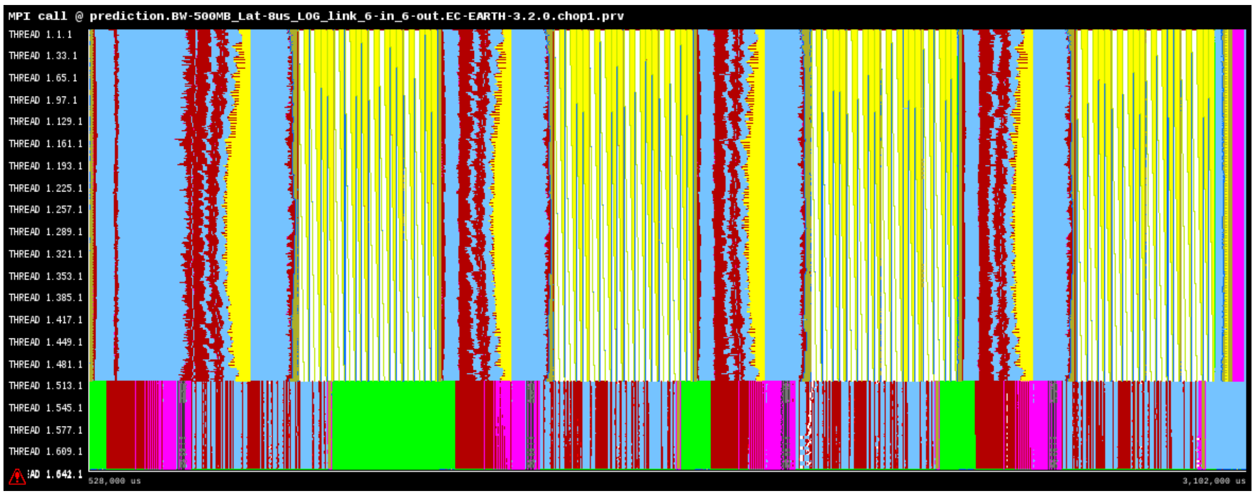


*Figure 21: Simulated trace with Dimemas base configuration*

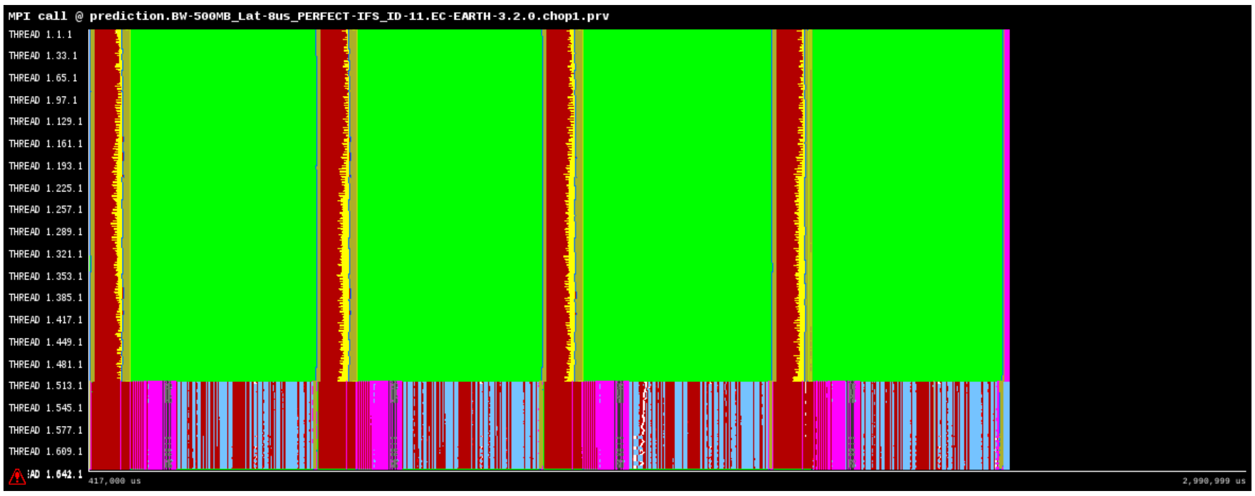


*Figure 22: Simulated trace with IFS "disabled"*

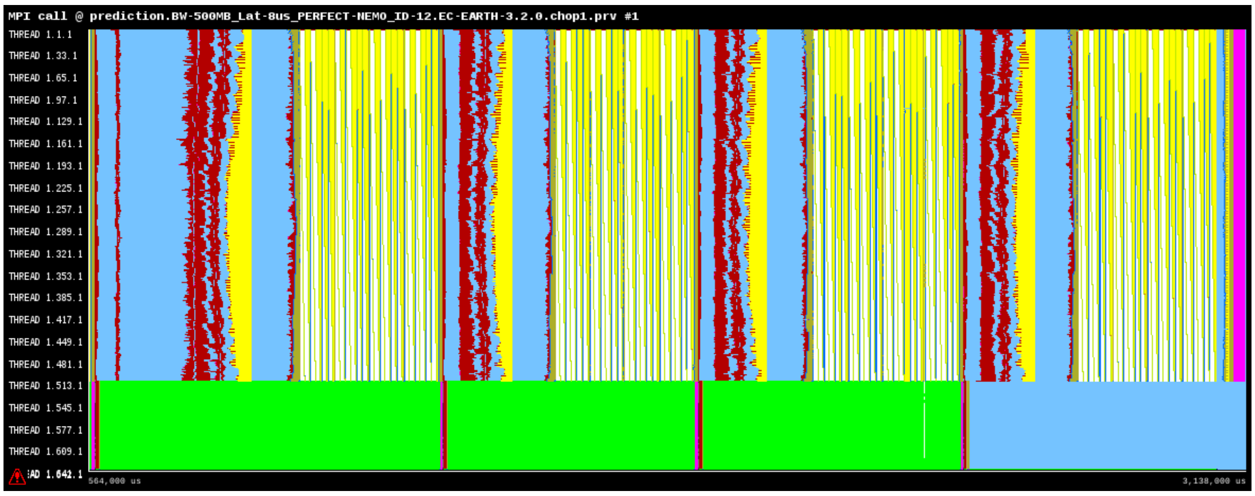


*Figure 23: Simulated trace with NEMO "disabled"*

# 6. Conclusion

In this document, we presented different simulated scenarios to make a performance analysis of EC-Earth using Dimemas tool, taking into account different parameters like processor speed, network latency and network bandwidth. We simulated three scenarios:

- Ideal machine: an execution has been simulated using no latency and infinite bandwidth.
- Model sensitivity: several executions have been simulated using infinite processor speed, different values for latency and different values for bandwidth.
- Limiting model due to coupling: two executions have been simulated by “disabling” IFS or NEMO at each execution. By “disabling” is meant to make one model much faster than usual to see how affects to the other one. Thus, it is possible to see which is the slower model of EC-Earth.

By simulating these scenarios, we realized that each model has its own characteristics. With the ideal machine, we have seen that there are several sources of inefficiency: about a 20.59% of the execution time is communication; there are workload imbalances between IFS and NEMO at each time step, but also within each model; and there are data dependencies.

In the model sensitivity simulations, we have described the types of messages and detected data dependencies. In IFS we have seen that latency affects the conservative part because there are dependent small messages, while bandwidth affects another region of the code with few big messages.

In NEMO, although it is known that is not very efficient, the simulated latencies and bandwidths in this report only affect slightly to its execution time. However, it has data dependencies and workload imbalances.

The last simulation test performed to detect the slowest model due to coupling, has revealed that using 512 MPI processes for IFS and 128 for NEMO, IFS is slower than NEMO. Moreover, there is not enough bandwidth to transfer all the data in IFS, whereas in NEMO there is almost no contention.

These results can not only help to detect bottlenecks of EC-Earth that need to be improved, but also to help code developers to design new future algorithms more machine-independent.

## Acknowledgements

This work has been supported by the IS-ENES2 project, funded by the European Union under grant agreement FP7 GA 312979.

## References

[1] Hazeleger W., X. Wang, C. Severijns, S. Ştefănescu, R. Bintanja, A. Sterl, K. Wyser, T. Semmler, S. Yang, B. van den Hurk, T. van Noije, E. van der Linden, K. van der Wiel, 2011: "EC-Earth V2.2: description and validation of a new seamless earth system prediction model". Clim Dyn, doi:10.1007/s00382-011-1228-5

[2] Yepes-Arbós, X., M.C. Acosta, K. Serradell, O. Mula-Valls, F.J. Doblas-Reyes, 2016: "Scalability and performance analysis of EC-Earth 3.2.0 using a new metric approach (Part II)". Barcelona Supercomputing Center

[3] Acosta, M.C., X. Yepes-Arbós, S. Valcke, E. Maisonnave, K. Serradell, O. Mula-Valls, F.J. Doblas-Reyes, 2016: "Performance analysis of EC-Earth 3.2: Coupling". Barcelona Supercomputing Center

[4] Barcelona Supercomputing Center. BSC performance tools, 2016: https://tools.bsc.es/